Metaeuristici care folosesc o soluție candidat (II)

- Simulated Annealing
- Tabu Search
- Variable Neighborhood Search
- Guided Local Search
- GRASP = Greedy Randomized Adaptive Search Procedure

Simulated Annealing

Idee:

- se acceptă, cu o anumită probabilitate, și modficări ale configurației curente care conduc la creșterea funcției obiectiv

Sursa de inspirație:

- procesul de reorganizare a structurii unui solid supus unui tratament termic:
  - Solidul este încălzit (topit): particulele sunt distribuite într-o manieră aleatoare
  - Solidul este răcit lent: particulele se reorganizează pentru a se ajunge la configurații de energie din ce în ce mică

Terminologie:

simulated annealing = tratament termic simulat = călire simulată

Istoric: Metropolis(1953), Kirkpatrick, Gelatt, Vecchi (1983), Cerny (1985)

Simulated Annealing

Analogie:

Proces fizic:		Problemă de minimizare:
• Energia sistemului	→	Funcție obiectiv
• Starea sistemului	→	Configurație (soluție candidat)
• Modificarea stării sistemului	→	Perturbarea configurației curente
• Temperatura	→	Parametru de control a procesului de optimizare

SA= metoda euristică inspirată de procese fizice

Simulated Annealing

Puțină fizică:

- fiecare stare a unui sistem este caracterizată de o anumită probabilitate de apariție

- probabilitatea asociată unei stări depinde de energia stării și de temperatura sistemului (ex: distribuția Boltzmann)

$$P_T(s) = \frac{1}{Z(T)} \exp(-\frac{E(s)}{k_B T})$$

$$Z(T) = \sum_{s \in S} \exp(-\frac{E(s)}{k_B T})$$

E(s) = energia stării s
T = temperatura sistemului
k_B = constanta Boltzmann
Z(T)=funcția de partiție
(factor de normalizare)

Simulated Annealing

Puțină fizică:

- Valori mari ale lui T (T tinde la infinit): argumentul lui exp este aproape 0 => stările sunt echiprobabile

- Valori mici ale lui T (T tinde la 0): vor avea probabilitate nenulă doar stările cu energia nulă

$$P_T(s) = \frac{1}{Z(T)} \exp(-\frac{E(s)}{k_B T})$$

$$Z(T) = \sum_{s \in S} \exp(-\frac{E(s)}{k_B T})$$

E(s) = energia stării s
T = temperatura sistemului
k_B = constanta Boltzmann
Z(T)=funcția de partiție
(factor de normalizare)

Simulated Annealing

Cum folosim acest lucru pentru rezolvarea unei probleme de optimizare ?

- Ar fi suficient să generăm configurații în conformitate cu distribuția Boltzmann pentru valori din ce în ce mai mici ale temperaturii

- **Problema:** dificil de calculat Z(T) (presupune calculul unei sume pentru toate stările posibile, adică generarea tuturor configurațiilor spațiului de căutare - IMPOSIBIL de realizat practic dacă spațiul configurațiilor este mare)

- **Soluție:** se aproximează distribuția prin simularea evoluției unui proces stohastic (lanț Markov) a cărui distribuție staționară coincide cu distribuția Boltzmann => algoritmul Metropolis

Simulated Annealing

Algoritmul Metropolis (1953)

s(0)=aproximație inițială

k=0

REPEAT

s'=perturb(s(k))

IF f(s')<f(s(k)) THEN s(k+1)=s' (necondiționat)

ELSE s(k+1)=s'

cu probabilitatea exp(-(f(s')-f(s(k))/T)

k=k+1

UNTIL "este indeplinita o conditie de terminare"

Obs: o noua configurație (s') este acceptată cu probabilitatea cu min{1,exp(-(f(s')-f(s(k))/T)}

Simulated Annealing

Algoritmul Metropolis – proprietăți

- Altă probabilitate de acceptare:

  $P(s(k+1)=s') = 1/(1+\exp((f(s')-f(s(k))/T))$

- Implementarea unei reguli de atribuire cu o anumită probabilitate

  u=rand(0,1)

  IF u<P(s(k+1)=x') THEN s(k+1)=s'

  ELSE s(k+1)=s(k)

- Valori mari pentru T -> probabilitate mare de acceptare a oricarei configurații (similar cu căutarea pur aleatoare)

  Valori mici pentru T -> probabilitate mare de acceptare doar pentru configurațiile care conduc la micșorarea funcției obiectiv (similar cu o metodă de descreștere sau căutare de tip greedy)

Simulated Annealing

Algoritmul Metropolis – proprietăți

- Generarea unor noi configurații depinde de problema de rezolvat

Optimizare în domenii continue

s'=s+z

$s=(z_1,\ldots,z_n)$

z_i : generata in cf. cu repartitia

- N(0,T)
- Cauchy(T) (Fast SA)
- altele

Optimizare combinatorială

Noua configurație se selectează determinist/aleator din vecinătatea configurației curente

Exemplu: TSP – transformare de tip 2-opt

Simulated Annealing

Simulated Annealing = aplicare repetată a algoritmului Metropolis pentru valori din ce în ce mai mici ale temperaturii

Structura generală

Init s(0), T(0)

i=0

REPEAT

  aplică Metropolis (pentru una sau mai multe iteratii)

  calcul T(i+1)

  i=i+1

UNTIL T(i)<eps

Problema: alegerea schemei de modificare a temperaturii (“cooling scheme”)

Simulated Annealing

Scheme de răcire:

$T(k)=T(0)/(k+1)$

$T(k)=T(0)/\ln(k+c)$

$T(k)=aT(k-1)$ $(a<1$, ex: $a=0.995)$

Obs. 1. T(0) se alege astfel încât la primele iterații să fie acceptate aproape toate configurațiile generate (pentru a asigura o bună explorare a spațiului soluțiilor)

2. Pe parcursul procesului iterativ este indicat să se rețină de fiecare dată cea mai bună valoare întâlnită

Simulated Annealing

Proprietăți de convergență:

Dacă sunt satisfăcute proprietățile:

- Pg(s(k+1)=s'|s(k)=s)>0 pentru orice s și s' (probabilitatea de trecere între oricare două configurații este nenulă)
- Pa(s(k+1)=s'|s(k)=s)=min{1,exp(-(f(s')-f(s))/T)} (probabilitate de acceptare de tip Metropolis)
- T(k)=C/lg(k+c) (schema logaritmică de răcire)

Atunci P(f(s(k))=f(s*)) -> 1 (s(k) tinde în probabilitate la minimul global s* când k tinde la infinit)

Simulated Annealing

Variante: alte probabilități de acceptare (Tsallis)

$$P_a(s') = \begin{cases} 1, & \Delta f \le 0 \\ (1-(1-q)\Delta f / T)^{1/(1-q)}, & \Delta f > 0, (1\text{-}q)\Delta f \le 1 \\ 0, & \Delta f > 0, (1\text{-}q)\Delta f > 1 \end{cases}$$

$$\Delta f = f(s') - f(s)$$

$$q \in (0,1)$$

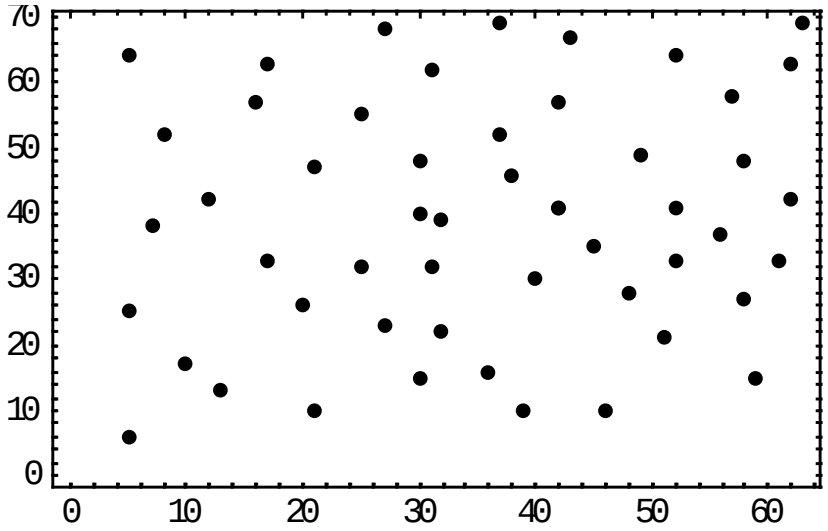
Simulated Annealing

Exemplu: problema comis voiajorului
(TSPLib: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95)
Instanta test: eil51 – 51 orase

Parametrii:

- 5000 iterații cu modificarea parametrului T la fiecare 100 de iterații
- T(k)=T(0)/(1+log(k))

Distribuția oraselor

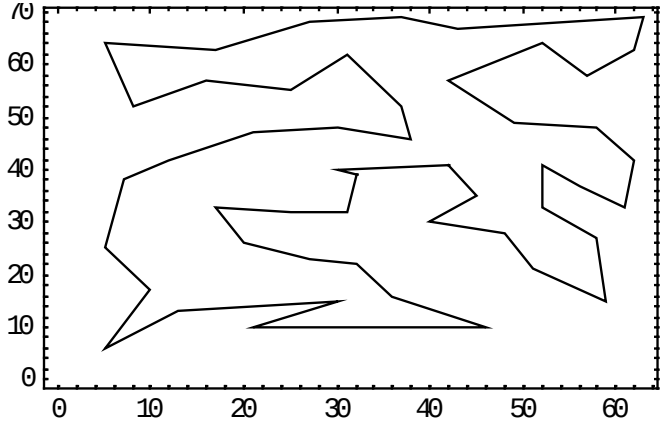


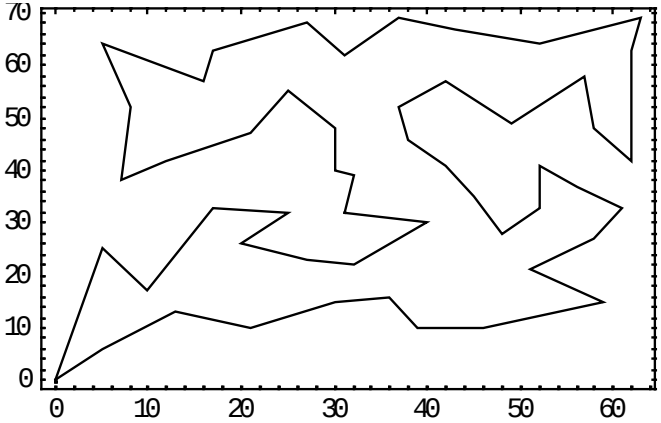
Simulated Annealing

Exemplu: problema comis voiajorului

Instanta test: eil51 (TSPlib)

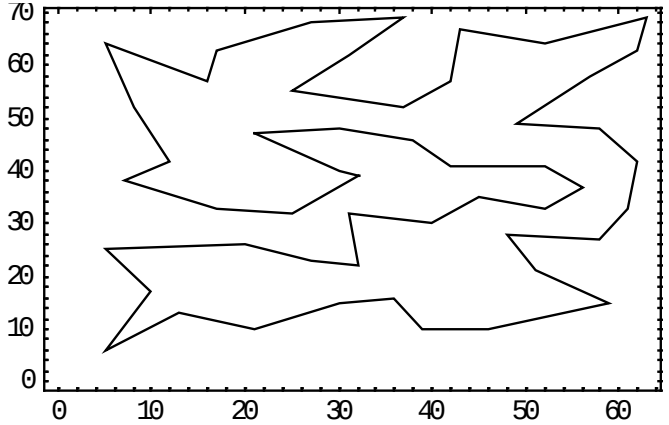
T(0)=5, cost=474.178





T(0)=10,cost=478.384

T(0)=1, cost=481.32



Cost minim: 426

Tabu Search

Căutare bazată pe liste de configurații interzise (TS – Tabu Search)
Creator: Fred Glover (1986)

Scop: rezolvarea problemelor de optimizare combinatorială

Specific:

- Tehnică iterativă de căutare locală bazată pe explorarea vecinătății configurației curente (vecinătatea unei configurații se definește ca fiind mulțimea configurațiilor ce pot fi atinse din configurația curentă printr-o singură transformare; transformările posibile sunt specifice problemei)

- Utilizează o listă de configurații interzise care nu vor putea fi vizitate în următoarele iterații; lista tabu are o dimensiune limitată (e implementată ca listă circulară)

Tabu Search

Căutare bazată pe liste de configurații interzise (TS – Tabu Search) Structura generala:

Pas 1: se construiește o configurație inițiala

Pas 2: REPEAT

- Se selectează cel mai bun element din vecinătatea configurației curente care este acceptabil în raport cu lista tabu
- Dacă elementul selectat este suficient de bun atunci se adaugă la o arhivă cu elite
- Se actualizează lista tabu

UNTIL <conditie de oprire>

Obs:

1. Dacă vecinătatea unei configurații este prea mare atunci nu se evaluează toate elementele ci doar o selecție a acestora
2. Pentru eficientizarea lucrului cu lista tabu în loc de configurații se pot stoca caracteristici ale acestora sau descrieri ale transformărilor efectuate

Tabu Search

Pentru eficientizarea lucrului cu lista tabu în loc de configurații se pot stoca caracteristici ale acestora sau descrieri ale transformărilor efectuate

Exemple de caracteristici:

- In cazul unei reprezentări de tip permutare interschimbarea a două elemente poate reprezenta o astfel de caracteristică. De exemplu (i,j) specifică faptul că elementul de pe pozitia i a fost interschimbat cu elementul de pe pozitia j
- In cazul unei reprezentări de tip vector binar complementarea valorii unei componente poate reprezenta o caracteristică. De exemplu (i,0) reprezintă faptul că pe poziția i este plasată valoarea 0, iar (i,1) faptul că pe poziția i este plasată valoarea 1.

Obs: utilizarea caracteristicilor în loc de liste complete poate conduce la situații în care se restricționează căutarea prin excluderea unor configurații foarte bune. Pentru a evita astfel de situații se permite acceptarea unor configurații bune chiar dacă corespund unor caracteristici trecute în lista tabu.

Tabu Search

Pentru îmbunătățirea funcționării se pot aplica periodic etape de intensificare și diversificare a căutarii

Intensificare:

Scop: exploatarea regiunilor ce par promițătoare

Mod de implementare:

- Se contorizează pentru fiecare componentă a configurației curente numărul de iterații consecutive în care a rămas nemodificată
- Se restartează procesul de căutare pornind de la cea mai bună configurație întâlnită și considerând fixate componentele cu comportare bună (pentru care contorul are valori mari)

Alte variante

Pentru îmbunătățirea funcționării se pot aplica periodic etape de intensificare și diversificare a căutarii

Diversificare:

Scop: explorarea regiunilor ce nu au fost vizitate

Mod de implementare:

- Se contorizează frecvența de utilizare, pe parcursul întregului proces iterativ, a valorilor corespunzătoare diferitelor componente
- Se restartează procesul de căutare de la configurații în care sunt plasate componente cu frecvența mică de utilizare; sau se penalizează scorul unei configurații folosind frecvențele asociate componentelor; relaxarea restricțiilor prin modificarea sistematică (creștere urmată de descreștere și invers) a ponderilor utilizate în funcția obiectiv care include restricțiile (construită prin tehnica penalizării)

Variable Neighborhood Search

Căutare bazată pe vecinătăți variabile (VNS - Variable Neighborhood Search)
[Mladenovic, P. Hansen, 1997]

- Idee: utilizează o structură de vecinătăți V_1, V_2,...,V_{kmax} care este explorată incremental; in cadrul fiecarei vecinătăți căutarea se realizează utilizând o metodă de căutare locală
- Obs: Structura de vecinătăți ale unei configurații (soluții candidat) x se stabilește în funcție de specificul problemei dar astfel încât dacă k1<k2 atunci elementele lui $V_{k1}(x)$ sunt mai „apropiate" de x decât elementele lui $V_{k2}(x)$
- Exemple:
  - pentru problema comis voiajorului $V_k(x)$ poate conține configurațiile obținute din x aplicând k interschimbări de noduri (locații) selectate aleator
  - Pentru problema rucsacului $V_k(x)$ poate conține configurațiile obținute din x prin modificarea a k componente

Variable Neighborhood Search

Căutare bazată pe vecinătăți variabile (VNS - Variable Neighborhood Search) [Mladenovic, Hansen, 1997]

Structura generală

Inițializează x (aleator în spațiul de căutare)

k=1

WHILE k<=kmax DO

selectează x' aleator din $V_k(x)$;

construiește x'' din x' aplicând o metodă de căutare locală (care nu e obligatoriu să se limiteze la $V_k(x)$)

IF f(x'')<f(x) THEN x=x''; k=1

ELSE k=k+1

Variable Neighborhood Search

Variable Neighborhood Decomposition Search

- Elementul cheie al algoritmilor de tip VNS este alegerea vecinătăților (ceea ce depinde de problema de rezolvat) – acestea controlează structura spațiului de căutare
- O variantă de definire a vecinătăților este cea în care se fixează o parte din atribute (în felul acesta problema inițială se descompune în subprobleme de dimensiune mai mică)
- Numărul de componente fixate este corelat cu dimensiunea vecinătății (de exemplu $V_k(x)$ conține elemente care diferă de x în maxim k componente, celelalte n-k componente fiind fixate). In procesul de căutare locală componentele fixate nu se modifică.

Guided Local Search

Guided Local Search = căutare locală ghidată [Voudouris&Tsang, 1999]

Idee: modificarea dinamică a funcției obiectiv pentru a permite evadarea din minime locale (prin alterarea valorilor asociate funcției obiectiv optimele locale deja vizitate devin mai puțin „dezirabile”)

Modificarea se bazează pe analiza prezenței unor caracteristici în soluția evaluată (de exemplu pt problema comis voiajorului o muchie poate fi considerată drept caracteristică)

$$f_{\text{mod}}(S) = f(S) + \lambda \sum_{i=1}^{m} p_i I_i(S)$$

$$I_i(S) = \begin{cases} 1 & \text{caracteristica } i \text{ este prezenta in } S \\ 0 & \text{caracteristica } i \text{ nu este prezenta in } S \end{cases}$$

$p_i = \text{parametru de penalizare}$

$\lambda = \text{factor de regularizare (controleaza impactul penalizarii)}$

Guided Local Search

Guided Local Search = căutare locală ghidată
[Voudouris&Tsang, 1999]

Structura generală:

S=configurație inițială
Repeat
  s=LocalSearch(S,f)
  for <all features i with maximal utility U(s,i)>
    $p_i=p_i+1$
  endfor
  f=Update(f,p)
until <conditie de oprire>

$$U(s,i) = I_i(s)\frac{c_i}{1+p_i}$$

$$f_{\text{mod}}(S) = f(S) + \lambda\sum_{i=1}^{m} p_i I_i(S)$$

$$I_i(S) = \begin{cases} 1 & \text{caracteristica } i \text{ este prezenta in } S \\ 0 & \text{caracteristica } i \text{ nu este prezenta in } S \end{cases}$$

p_i = parametru de penalizare

λ = factor de regularizare (controleaza impactul penalizarii)

GRASP

GRASP = Greedy Randomized Adaptive Search Procedure

Idee: configurația inițială este construită folosind

- O euristică care permite construirea componentă cu componentă a configurației
- Alegerea valorii corespunzătoare fiecărei componente se face prin selecție aleatoare dintr-o listă ierarhizată de valori posibile
- Dimensiunea listei este variabilă; cazuri extreme:
  - Dimensiunea = 1 -> alegere greedy (se selectează cea mai bună componentă)
  - Dimensiunea = nr maxim de valori posibile pt o componentă -> căutare pur aleatoare